Journal of Machine Learning Research 1 (2009) 1-4 Submitted 5/09; Published -/-

DL-Learner: Learning Concepts in Description Logics

Jens Lehmann LEHMANN@INFORMATIK.UNI-LEIPZIG.DE
Department of Computer Science
University of Leipzig
Johannisgasse 26, 04103 Leipzig, Germany

Editor: –

Abstract

In this paper, we introduce DL-Learner, a framework for learning in description logics and OWL. OWL is the official W3C standard ontology language for the Semantic Web. Concepts in this language can be learned for constructing and maintaining OWL ontologies or for solving problems similar to those in Inductive Logic Programming. DL-Learner includes several learning algorithms, support for different OWL formats, reasoner interfaces, and learning problems. It is a cross-platform framework implemented in Java. The framework allows easy programmatic access and provides a command line interface, a graphical interface as well as a WSDL-based web service.

Keywords: Concept Learning, Description Logics, OWL, Classification, Open-Source

1. Introduction

The *Semantic Web* grows steadily[1] and contains knowledge from diverse areas such as science, music, literature, geography, social networks, as well as from upper and cross domain *ontologies*[2]. The underlying semantic technologies currently start to create substantial industrial impact in application scenarios on and off the web, including knowledge management, expert systems, web services, e-commerce, e-collaboration, etc. Since 2004, the Web *Ontology Language OWL*, which is based on *description logics* (Baader et al., 2007), has been the W3C-recommended standard for Semantic Web ontologies and is a key to the growth of the Semantic Web.

Within this field, there is a need for well-structured ontologies with large amounts of instance data, since engineering such ontologies constitutes a considerable investment of resources. Nowadays, knowledge bases often provide large amounts of instance data without sophisticated schemata. Methods for automated schema acquisition and maintenance are therefore sought (see e.g. Buitelaar et al. (2007)). In particular, concept learning methods have attracted much interest, see e.g. Esposito et al. (2004); Lehmann (2007); Lehmann and Hitzler (2008); Lisi and Esposito (2008). DL-Learner provides an open source framework for such methods as we will briefly describe in the sequel. Several learning algorithms have been implemented within this framework. Outside of DL-Learner, there exist only non open source implementations of algorithms (YinYang, DL-FOIL) to the best of our knowledge.

1. To give a rough estimate, the semantic index Sindice (http://sindice.com/) lists more than 10 billion entities from more than 100 million web pages.
2. See e.g. `http://tomgruber.org/writing/ontology-definition-2007.htm` for a definition of ontology in computer science.

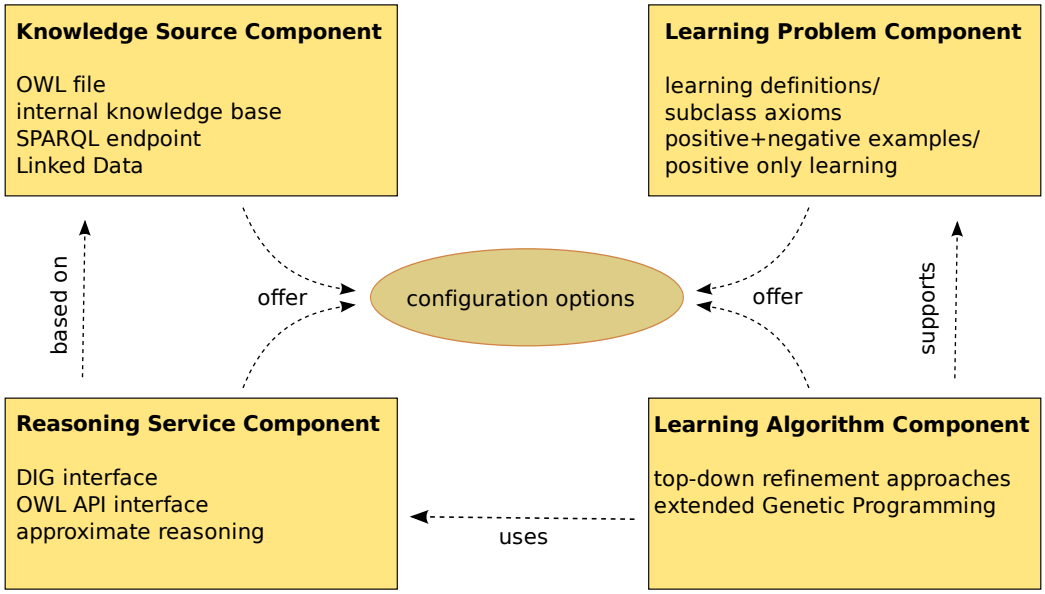

Figure 1: The architecture of DL-Learner is based on four component types each of which can have their own configuration options. A component manager can be used to create, combine, and configure components.

2. Framework

DL-Learner consists of core functionality, which provides Machine Learning algorithms for solving learning problems in OWL, support for different knowledge base formats, an OWL library, and reasoner interfaces. There are several interfaces for accessing this functionality, a couple of tools which use the DL-Learner algorithms, and a set of convenience scripts.

To be flexible and easily extensible, DL-Learner uses a component-based model. There are four types of components: knowledge source, reasoning service, learning problem, and learning algorithm. For each type, there are several implemented components and each component can have its own configuration options as illustrated in Figure 1. Configuration options can be used to change parameters/settings of a component.

Knowledge Sources integrate background knowledge. Almost all standard OWL formats are supported through the OWL API[3], e.g. RDF/XML, Manchester OWL Syntax, or Turtle. DL-Learner supports the inclusion of several knowledge sources, since knowledge can be widespread in the Semantic Web. In addition, DL-Learner facilitates the extraction of knowledge fragments from SPARQL[4] endpoints. This feature allows DL-Learner to scale up to very large knowledge bases containing millions of axioms cf. Hellmann et al. (2009).

Reasoner Components provide connections to existing or own reasoners. Two components are the DIG 1.1[5] and OWL API reasoner interfaces, which allow to connect to all standard OWL reasoners via an HTTP and XML-based mechanism or a Java interface,

3. http://owlapi.sourceforge.net
4. http://www.w3.org/TR/rdf-sparql-query/
5. http://dl.kr.org/dig/

respectively. Furthermore, DL-Learner offers its own approximate reasoner, which uses Pellet[6] for bootstrapping and loading the inferred model in memory. Afterwards, instance checks are performed very efficiently by using a local closed world assumption (see Badea and Nienhuys-Cheng (2000) on why this assumption is useful in description logics).

Learning Problems specify the problem type, which is to be solved by an algorithm. Currently, three problem components are implemented: 1.) learning from positive and negative examples 2.) positive-only learning and 3.) class axiom learning. The latter type is split into learning definitions and super class axioms. Amongst other methods, the components provide efficient coverage checks which can be used in the learning algorithms, e.g. stochastic approaches for computing coverage up to a desired accuracy with respect to a 95% confidence interval are available.

Learning Algorithm components provide methods to solve one or more specified learning problem types. Apart from simple algorithms involving brute force or random guessing techniques, DL-Learner comprises a number of sophisticated algorithms based on genetic programming with a novel genetic operator (Lehmann, 2007), refinement operators for the description logic $\mathcal{ALC}$ (Lehmann and Hitzler, 2008), an extended operator supporting many features of OWL including datatype support, and an algorithm tailored for ontology engineering with a strong bias on short and readable concepts. Some of those algorithms have shown to be superior to other description logic learning systems and also superior to state-of-the-art ILP systems, e.g. on the carcinogenesis problem[7].

3. Implementation

The homepage of DL-Learner is http://dl-learner.org and contains up-to-date information about documentation and development of the software. A manual[8], which complements the homepage and describes how to run DL-Learner, is included in its release. For developers, the Javadoc of DL-Learner is available online[9].

The code base of DL-Learner consists of approximately 50,000 lines of code (excluding comments) with its core, i.e. the component framework itself, accounting for roughly 1,500 lines. It is licensed under GPL 3. About 20 learning examples are included in the latest release (to be precise: 132 if smaller variations of existing problems/configurations are counted). 27 unit tests based on the JUnit framework are used to detect errors.

There are several interfaces available to access DL-Learner: To use components programmatically, the core package, in particular the component manager, can be of service. Similar methods are also available at the web service interface, which is based on WSDL. DL-Learner starts a web service included in Java 6, i.e. no further tools are necessary. For end users, a command line interface is available. Settings are stored in *conf files*, which can then be executed in a similar fashion to other ILP tools. A prototypical graphical user interface is equally available, which can create, load, and save conf files. It provides widgets for modifying components and configuration options. An advantage of the component-

6. `http://clarkparsia.com/pellet/`
7. `http://dl-learner.org/wiki/Carcinogenesis` presents benchmark results
8. `http://dl-learner.org/files/dl-learner-manual.pdf`
9. `http://dl-learner.org/javadoc/`

based architecture is that all the interfaces mentioned need not to be changed, when new components are added or existing ones modified. This makes DL-Learner easily extensible. Another means to access DL-Learner, in particular for ontology engineering, is through plugins for the ontology editors OntoWiki[10] and Protégé[11]. The OntoWiki plugin is under construction, but can be used in its latest SVN version. The Protégé 4 plugin is included in the official Protégé plugin repository, i.e. it is easy to install within Protégé.

Acknowledgments

I wish to acknowledge support by the OntoWiki EU FP7 project. Special thanks goes to Francesca Lisi for her comments, as well as the developers working on DL-Learner and tools based on it. Acknowledgements particularly to Sebastian Hellmann who worked on the SPARQL component, to Christoph Haase for his work on an EL refinement operator, to Christian Kötteritzsch for his work on the Protege plugin, to Sebastian Bader who contributed a Prolog parser, and to those people using DL-Learner in their software.

References

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2007. Cambridge University Press. ISBN 0-521-78176-0.

L. Badea and S.-H. Nienhuys-Cheng. A refinement operator for description logics. In *Proc. of the 10th Int. Conf. on Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 40–59. Springer-Verlag, 2000.

P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence*. IOS, 2007.

F. Esposito, N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro. Knowledge-intensive induction of terminologies from metadata. In *Proc. of 3rd Int. Semantic Web Conf.*, pages 441–455. Springer, 2004.

S. Hellmann, J. Lehmann, and S. Auer. Learning of OWL class descriptions on very large knowledge bases. *Int. Journal On Semantic Web and Information Systems*, 2009.

J. Lehmann. Hybrid learning of ontology classes. In *Proc. of 5th Int. Conf. on Machine Learning and Data Mining in Pattern Recognition*, volume 4571 of *Lecture Notes in Computer Science*, pages 883–898. Springer, 2007.

J. Lehmann and P. Hitzler. A refinement operator based learning algorithm for the ALC description logic. In *Proc. of 17th Int. Conf. on Inductive Logic Programming*, volume 4894 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2008. Awarded.

F.A. Lisi and F. Esposito. Foundations of onto-relational learning. In *Proc. of 18th Int. Conf. on Inductive Logic Programming*, volume 5194 of *Lecture Notes in Computer Science*, pages 158–175. Springer, 2008.

10. `http://ontowiki.net`
11. `http://protege.stanford.edu`